Divide and Conquer Attack on ABC Stream Cipher

Shahram Khazaei
Sharif University of Technology, Tehran, Iran
khazaaei@yahoo.com
July 2005

Abstract

ABC is a synchronous stream proposed as a candidate to ECRYPT Project. ABC gets a 128-bit key and a 128-bit IV and produces 1195 bits as the internal state of the cipher. Using some statistical simulations we show that one of the ABC components, a key-IV dependent function over $GF(2^{32})$ called C which is chosen randomly from a family of functions, is slightly better than a randomly chosen function over $GF(2^{32})$. Using this great weakness of C we propose a correlation based divide and conquer attack which finds 63 bits of the state bits by searching over all 2^{63} possible choices for them in time complexity of 10×2^{95} simple word operations using 10×2^{32} output words. Although the attack is performable using less data and time complexities, we have not considered it here. After we find these 63 bits of the initial state, searching over all 2^{93} possible choices for another set of 93 bits of the initial state and constructing a 33-unknown linear system of equations, we show that the whole initial state could be found in time complexity of 2^{108} simple word operations using a few output words. Therefore the total time and data complexity of our attack for breaking the whole cipher are 2^{108} simple word operations and 10×2^{32} words respectively.

Keywords. ABC stream cipher, ECRYPT Project, divide and conquer attack, correlation attack, hypothesis testing.

1 Introduction

ABC is a synchronous stream cipher optimized for software applications. ABC deals with a 128-bit key and a 128-bit IV proposed as a candidate to ECRYPT Stream Cipher Project- a multi-year effort to identify new stream ciphers that might become suitable for widespread adoption [2]. ABC consists of 38, 32-bit registers which three of them denoted by $\mathbf{z}^0$, $\mathbf{z}^1$, $\mathbf{x}$ are considered as the state of ABC cipher and the rest 35 registers, denoted by $\mathbf{d}_0$, $\mathbf{d}_1$, $\mathbf{e}$, $\mathbf{e}_0$, $\mathbf{e}_1$, … and $\mathbf{e}_{31}$, are considered as constant parameters fed to the cipher. The values of these 38 registers are determined during a key expansion routine. Although each register is a 32-bit word, the total number of initial bits are 1195 bits and not 1216 bits because of restrictions $\mathbf{z}^0 \equiv 2$ or 3 (mod 4), $\mathbf{e}_{31} \equiv 2^{16} \pmod{2^{17}}$, $\mathbf{d}_0 \equiv 1$ (mod 2) and $\mathbf{d}_1 \equiv 0$ (mod 4) which are made during initialization. The main components of ABC are three filter functions denoted by A, B and C. The designers of ABC have claimed that ABC offers a security level of 2^{128}, and C function is its main security block. In this paper we show that C is far from a random substitution box and more likely

behaves as a randomly chosen function which is not acceptable for cryptographic purposes. Using this weakness of C we propose a divide and conquer correlation attack on ABC targeting the 2^{63} possible initial states of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers. Our analysis shows that the initial states of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers can be found using 10×2^{32} output words. Although the attack is performable using less output words, we have not considered it here. Since for each 2^{63} possible initial state of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers we have to process 10×2^{32} words, the time and data complexities of this phase are 10×2^{95} simple word operations and 10×2^{32} words respectively.

After finding the initial state of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers, searching over all 2^{93} possible initial values for $\mathbf{x}$, $\mathbf{d}_0$ and $\mathbf{d}_1$ registers, the values of $\mathbf{e}$, $\mathbf{e}_0$, $\mathbf{e}_1$, … and $\mathbf{e}_{31}$ could be found by constructing a linear system of equations versus these 33 unknowns for each guess. This phase needs 2^{108} simple word operations and a few words of the output sequence. Therefore the total time and data complexity of our attack for breaking the whole cipher are 2^{108} simple word operations and 2^{32} words, respectively.

The paper is organized as follows. In section 2, a brief description of ABC components and key stream generator algorithm is presented. In section 3 the non-uniform distribution of C family function is discussed. In section 4 we will explain how to distinguish the output sequence of a purely random filtered sequence from a purely random sequence where the filter function is chosen randomly from a set of functions. The attack procedure to determine the initial state of ($\mathbf{z}^0$, $\mathbf{z}^1$) registers and break the whole cipher is presented in Section 5. Conclusions are given in Section 6.

2 ABC Stream Cipher Description

In this section we briefly describe the ABC key stream generator algorithm and components. For more details on ABC and its key schedule see the reference paper [1].

2.1 ABC Components

ABC works with 32-bit integer values. A 32-bit vector (a_{31},a_{30},...,a_1,a_0) is denoted by integer $\mathbf{a}$ where $\mathbf{a} = \sum_{i=0}^{31} a_i 2^i$. ABC uses 38 32-bit variables that are calculated from the key and IV at the initialization stage by applying a special key expansion routine. These variables are saved in 38 registers denoted by $\mathbf{z}^0$, $\mathbf{z}^1$, **x,** **e**, $\mathbf{e}_0$, …, $\mathbf{e}_{31}$, $\mathbf{d}_0$ and $\mathbf{d}_1$. Having been defined once, the variables $\mathbf{d}_0$, $\mathbf{d}_1$, **e** and $\mathbf{e}_i$'s, remain unchanged during the whole subsequent encryption stage. The variables $\mathbf{z}^0$, $\mathbf{z}^1$ and **x** construct the current internal state of the ABC stream cipher and are updated while each 32-bit output word denoted by **y** is produced. Throughout this paper the symbols $\oplus, >>, <<, >>>$ and $<<<$ are respectively used for 32-bit xor, right shift, left shift, right rotation and left rotation, and the symbols $\cdot, +, -$ and $\sum$ are respectively used for multiplication, addition, subtraction and summation module 2^{32} unless otherwise stated. ABC key stream generator uses three filter functions A, B and C defined by the following expressions.

$$A(z^1, z^0) = (z^1 \oplus (z^0 >> 1) \oplus (z^1 << 31), z^1) \quad (1)$$

$$B(x) = d_0 + 5 \cdot (x \oplus d_0) \quad (2)$$

$$C(x) = (e + \sum_{i=0}^{31} x_i e_i) >>> 16 \quad (3)$$

A is a linear transformation on the vector space GF(2^{64}). A(z^1,z^0) has been defined by an LFSR of length 64 with characteristic polynomial $\varphi(\theta) = \psi(\theta)\theta$, where $\psi(\theta) = \theta^{63} + \theta^{31} + 1$ is primitive. The cycle length of this LFSR is $2^{63} - 1$, and not $2^{64} - 1$. The cycle length becomes 1 in case that the initial state ($\mathbf{z}^0$, $\mathbf{z}^1$) of A is either (0, 0) or (0, 1), so this values are eliminated during initialization by forcing the second LSB of $\mathbf{z}^0$ to 1 in ABC initialization process.

B is a T-function [3] and the parameters $\mathbf{d}_0$ and $\mathbf{d}_1$ must be chosen in a way that $\mathbf{d}_0 \equiv 1 \pmod 2$ and $\mathbf{d}_1 \equiv 0 \pmod 4$. These restrictions guarantee that B is a single cycle map.

It has been claimed that C is a highly non-linear mapping and is the main security block of ABC. As it will be shown in Section 3, it is disaster for ABC stream cipher and makes our divide and conquer correlation attack possible. The value of $\mathbf{e}_{31}$ is chosen according to $\mathbf{e}_{31} \equiv 2^{16} \pmod{2^{17}}$. This restriction provides long period, uniform distribution, and high linear complexity of output sequences [1].

2.2 ABC Key Stream Generator Algorithm

The key stream generation routine of ABC involves the primitives described in Section 2.1 and consists of 3 steps.

ABC KEY STREAM GENERATOR

INPUT: $\mathbf{z}^0$, $\mathbf{z}^1$, $\mathbf{x} \in GF(2^{32})$

$(\mathbf{z}^0, \mathbf{z}^1) \leftarrow A(\mathbf{z}^0, \mathbf{z}^1)$ (4)

$\mathbf{x} \leftarrow \mathbf{z}^1 + B(\mathbf{x})$ (5)

$\mathbf{y} \leftarrow \mathbf{z}^0 + C(\mathbf{x})$ (6)

OUTPUT: $\mathbf{z}^0$, $\mathbf{z}^1$, $\mathbf{x}$, $\mathbf{y} \in GF(2^{32})$

This routine generates the next 32 key stream bits, $\mathbf{y}$. The newly computed values $\mathbf{z}^0$, $\mathbf{z}^1$ and $\mathbf{x}$ form the input of the next iteration of the key stream generator routine.

Denoting the states sequence and the output key stream sequence by $\{z_n^1, z_n^0, x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=1}^{\infty}$, the key stream generation routine can be expressed by the following recursive equations

$$(z_n^1, z_n^0) = A(z_{n-1}^1, z_{n-1}^0) \quad (7)$$

$$x_n = z_n^1 + B(x_{n-1}) \tag{8}$$

$$y_n = z_n^0 + C(x_n) \tag{9}$$

for $n \geq 1$, where z_0^0, z_0^1 and x_0 are respectively the initial values of $\mathbf{z}^0$, $\mathbf{z}^1$ and $\mathbf{x}$ registers.[1]

3 Statistical Behavior of C Family Functions

In this section we investigate the behavior of the output of the C function assuming a uniform distribution for its input over GF(2^{32}). Because of good statistical properties of LFSR's output sequences, it is reasonable to think of $\{x_n\}$ as a purely random sequence. By *distribution of* C(**x**) we mean the probability mass function of random variable C(**x**) considering **x** as a uniform random variable over its domain. It seems hard to theoretically compute the distribution of C(**x**). Particularly, it must be noted that the distribution of C(**x**) completely depends on the values of **e** and $\mathbf{e}_i$'s. Let Ω_m denote the set of all functions over GF(2^m). To investigate the behavior of the distribution of C(**x**), in a generalized model we define C_m as a subset of Ω_m, which every $F \in C_m$ is of the form

$$F(x) = e + \sum_{i=0}^{m-1} x_i e_i \tag{10}$$

where m=2t is an even integer, **e** and $\mathbf{e}_i$'s are integer values over GF(2^m) conditioned on $\mathbf{e}_{m-1} \equiv 2^t \pmod{2^{t+1}}$, $\mathbf{x} = \sum_{i=0}^{m-1} x_i 2^i$ is the integer representation of the vector (x_{m-1}, x_{m-2}, ..., x_0), and the symbols + and Σ respectively represent the addition and summation module 2^m.

Ignoring the 16 bit right rotation of the C function in ABC doesn't put any restriction in our analysis, so we can consider it as a member of C_{32}. The number of functions in C_m is 2^{m^2+t-1}, but the number of possible choices for C in ABC is determined by key expansion routine. For a good key expansion routine, as is the case for proposed ABC to ECRYPT, it is determined by the total length of the key and IV, that is 2^{256}. We neglect this difference and treat the C function in ABC as a uniformly distributed function over all 2^{32^2+16-1} possible functions in C_{32} instead of 2^{256}. This is equivalent to consider **e** and $\mathbf{e}_i$'s as uniform independent random variables.

---

[1] Note that $z_n^0 = z_{n-1}^1$ for $n \geq 1$ and we could deal with one of the sequences $\{z_n^0\}_{n=0}^{\infty}$ and $\{z_n^1\}_{n=0}^{\infty}$ instead of both of them, for example the sequence $\{z_n^0\}_{n=0}^{\infty}$ satisfies the recursive equation $z_n^0 = z_{n-1}^0 \oplus (z_{n-2}^0 >> 1) \oplus (z_{n-1}^0 << 31)$ for $n \geq 2$ where z_0^0 and z_1^0 are respectively the initial values of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers.

Every component of a cipher must satisfy some criteria in order to prevent some attacks. In some cases, i.e. T-functions [3], some interesting criteria could be proved theoretically. Usually any function F: $GF(2^m) \rightarrow GF(2^n)$ ($m \geq n$) which is designed for cryptographic purposes, at least is considered to be balanced, that is for every $y \in GF(2^n)$ there are exactly 2^{m-n} $x \in GF(2^m)$ satisfying $F(x) = y$. If m=n this criterion is equivalent to that F is a permutation. Although a theoretical justification of cryptographic properties of a cipher elements are desirable, in some situations we could evaluated them using some statistical simulations. The designers of ABC have not neither evaluated C fuction theoretically nor using statistical simulations and just have designed C function to provide a provably minimum period for its output sequences [1]. Choosing C according to the general form of (10) is not sufficient to guarantee to be even approximately a permutation. In the following theorem the condition and probability which under, a function in C_m is a permutation is given.

Theorem 1. Every $F \in C_m$ is a permutation if and only if for every subset M of the set {0, 1, 2, ..., m-1}, we have

$$\sum_{i \in M} e_i \neq 0$$

where $\sum$ is performed on module 2^m. Moreover the probability that a randomly chosen $F \in C_m$ be a permutation is $\prod_{k=1}^{m-1} \left(1 - (2^{k+1} - 1)2^{-m}\right)$.

Using approximation $1 + x \approx e^x$ for $|x| << 1$, it could be shown that this probability is approximately equal to $e^{-1}2^{-m}$. The probability that a randomly chosen $F \in \Omega_m$ be a permutation is $2^m! / (2^m)^{2^m}$, which using Stirling's approximation is approximately equal to $e^{-2^m}\sqrt{2\pi 2^m}$. Although the asymptotical behavior of these two probabilities are not similar, we will show that the behavior of a randomly chosen function over C_m slightly deviates from a randomly chosen function from Ω_m. To this end we first define two characteristics denoted by **Q** vector and **Λ** square matrix for a randomly chosen function from Ω, a given subset of functions over $GF(2^m)$. Then we compare **Q** elements and **Λ** diagonal elements for a randomly chosen function from Ω_m and C_m using some statistical simulations. We use these parameters to theoretically analyze our distinguishing method between the output of C function and uniform distribution in Section 4.

Let for every $F \in \Omega_m$ and $k \in \{0,1,2...2^m\}$, T_k denote the proportion of outputs which have been repeated exactly k times in the output range of F. For example T_0 means that the number of $x \in GF(2^m)$ which have not been appeared in the output range of F, is $2^m T_0$. If F is a permutation $T_0=1$ and $T_k=0$ for $k>0$. If F is picked randomly from a given Ω, the T_k's are random variables. We use the notations **Q** and **Λ** respectively for average vector and covariance matrix of the vector random variable $[T_0\ T_1\ \ldots\ T_k]$. If we denote the average and variance of T_k by q_k and s_k^2, q_k's and s_k^2's construct the elements of **Q** and the diagonal elements of **Λ** respectively.

The computation of **Q** and **Λ** for a randomly chosen function from either C_m or Ω_m is interesting. It seems very hard to theoretically compute these values. In the Tables 1 and 2 we have estimated the average and variance of T_k for m=8:4:24 over 100 randomly chosen functions from C_m.

Table 1. Estimated values of q_k over 100 randomly chosen functions from C_m

	0	1	2	3	4	5	6	7	8	9
8	0.3782	0.3457	0.2004	0.0548	0.0173	0.0018	0.0017	0.0001	0.0000	0.0000
12	0.3752	0.3567	0.1868	0.0610	0.0158	0.0035	0.0007	0.0002	0.0000	0.0000
16	0.3655	0.3691	0.1867	0.0612	0.0144	0.0027	0.0004	0.0000	0.0000	0.0000
20	0.3668	0.3690	0.1845	0.0610	0.0151	0.0030	0.0005	0.0001	0.0000	0.0000
24	0.3688	0.3668	0.1837	0.0614	0.0155	0.0031	0.0005	0.0001	0.0000	0.0000

Table 2. Estimated values of s_k^2 over 100 randomly chosen functions from C_m (multiplied by 2^m)

	0	1	2	3	4	5	6	7	8	9
8	2.7347	5.0129	0.8514	0.3872	0.1482	0.0090	0.0137	0.0002	0.0000	0.0000
12	9.2381	11.8759	2.1642	0.3994	0.3693	0.0787	0.0104	0.0012	0.0001	0.0000
16	29.624	37.9064	6.1175	1.7604	1.1824	0.1717	0.0109	0.0006	0.0000	0.0000
20	237.51	256.575	52.7279	8.4157	9.5599	1.7994	0.1669	0.0093	0.0005	0.0000
24	1476.0	1656.06	287.4170	43.6064	56.6035	10.8645	0.9688	0.0540	0.0021	0.0001

In order to compare the behavior of a randomly chosen function from C_m with a randomly chosen one from Ω_m, in Tables 3 and 4 we have also estimated the average and variance of T_k for m=8:4:24 over 100 randomly chosen functions from Ω_m.

Table 3. Estimated values of q_k over 100 randomly chosen functions from Ω_m

	0	1	2	3	4	5	6	7	8	9
8	0.3681	0.3670	0.1861	0.0590	0.0163	0.0029	0.0006	0.0001	0.0000	0.0000
12	0.3663	0.3699	0.1838	0.0616	0.0149	0.0029	0.0005	0.0001	0.0000	0.0000
16	0.3679	0.3679	0.1839	0.0612	0.0154	0.0031	0.0005	0.0001	0.0000	0.0000
20	0.3676	0.3682	0.1841	0.0613	0.0153	0.0031	0.0005	0.0001	0.0000	0.0000
24	0.3655	0.3703	0.1852	0.0609	0.0148	0.0029	0.0005	0.0001	0.0000	0.0000

Table 4. Estimated values of s_k^2 over 100 randomly chosen functions from Ω_m (multiplied by 2^m)

	0	1	2	3	4	5	6	7	8	9
8	0.0985	0.2419	0.1321	0.0407	0.0129	0.0028	0.0007	0.0001	0.0000	0.0000
12	0.0867	0.2048	0.1030	0.0461	0.0128	0.0028	0.0005	0.0001	0.0000	0.0000
16	0.0954	0.2424	0.1035	0.0423	0.0098	0.0027	0.0005	0.0001	0.0000	0.0000
20	0.0870	0.1911	0.1251	0.0447	0.0112	0.0027	0.0004	0.0001	0.0000	0.0000
24	0.0534	0.2133	0.0960	0.0250	0.0049	0.0038	0.0006	0.0000	0.0000	0.0000

Looking into Tables 1 to 4 the following results seems to be true:

1. For each k, the q_k value for a randomly chosen function from C_m is approximately the same as a randomly chosen function from Ω_m (Tables 1 and 3).
2. For each k, the q_k value is a convergent sequence versus m for a randomly chosen function from both C_m and Ω_m, and they approximately converge to the same sequence; esp. T_0, $T_1 \rightarrow e^{-1} \cong 0.3679$ (Table 1 and 3).
3. For each k, while the s_k^2 value for a randomly chosen function from C_m differs from a randomly chosen function from Ω_m, they both approaches to zero when m increases. But the convergence rate for a randomly chosen function from Ω_m is faster than one from C_m (divided values of Tables 2 and 4 by 2^m).
4. At least for k<7, the $2^m s_k^2$ value converges to a non-zero value when m increases for a randomly chosen function from Ω_m, while it diverges for a randomly chosen function from C_m (Tables 2 and 4).
5. The value of $\sum_{k=0}^{K} kq_k$ approaches to 1 as both K and m increases for a randomly chosen function from both Ω_m and C_m.

More simulations show that the same results are true for all entries of **Λ.** That is they all exponentially approach to zero versus m for both cases, but the convergence rate for a randomly chosen function from Ω_m is faster than one from C_m. We will refer to these results in Section 5.

It seems that the computation of **Q** vector and **Λ** matrix for a randomly chosen function from Ω_m be much easier than that from C_m. This problem can be considered as a generalization of the so-called "Balls and Bins Problem", which is proposed as a open problem.

Generalized Balls and Bins Problem. We are throwing *n* balls into *n* bins randomly (i.e., for every ball we randomly and uniformly pick a bin from the *n* available bins, and place the ball in the bin picked). We define the random variable T_k as the number of bins

which contain exactly k balls. What are the asymptotic behavior of average value of T_k and covariance value of T_k and T_i versus n for each k and i?

4 Distinguishing a Purely Random Filtered Sequence From a Purely Random Sequence

In this section we assume that F is a purely random chosen function from Ω, a given subset of functions over GF(2^m). The aim of this section is to make a hypothesis testing between these two hypotheses, given a sequence $\{w_n\}_{n=1}^{N}$.

H_0: $\{w_n\}_{n=1}^{N}$ is a purely random sequence over GF(2^m).

H_1: $\{w_n\}_{n=1}^{N}$ is a filtered purely random sequence over GF(2^m) where the filter function has been chosen randomly from Ω, that is $w_n=F(x_n)$ where $\{x_n\}$ is a purely random sequence over GF(2^m) and F is a randomly chosen function from Ω.

We focus on $\Omega=\Omega_m$ and $\Omega=C_m$ and propose a method for distinguishing between these two hypotheses which works with arbitrary small error probability, in time, memory and data complexities of $O(2^m)$.

Let p_x denote $\Pr\{w_n=x\}$. Under hypothesis H_0 we have $p_x = 2^{-m}$ for all $x\in GF(2^m)$, while under hypothesis H_1 the values of p_x are not known because the filter function F is not known. If we knew the F function exactly, the values of p_x were available, and a Likelihood Ratio Test could be applied. In this case which we don't know F, although it is easy to show that there is not a UMP test for this hypothesis testing problem, the presence of a UMPI test is under question. The interesting readers are referred to [5] for more details on UMP and UPI tests. In this paper we concentrate on finding an ad-hoc solution for this hypothesis testing problem rather than finding a UMPI one which is sufficient for our cryptanalysis.

The ad-hoc statistic which we are going to use for this problem in some way is similar to Index of Coincidence (IC) or Measure of Roughness (MR) which is used in cryptanalysis of Classic Ciphers [6]. Let $\hat{f}_x$ and $\hat{p}_x$ respectively denote the number and frequency of occurrence of x in the sequence $\{w_n\}_{n=1}^{N}$ for each $x\in GF(2^m)$, that is $\hat{p}_x = \hat{f}_x / N$. For a given sequence $\{w_n\}$ we use the following statistic for distinguishing between H_0 and H_1.

$$\Delta = \sum_{x=0}^{2^m-1} | \hat{p}_x - 2^{-m} | \qquad (11)$$

We suppose that $N = \lambda 2^m$. In this paper we just consider the case $\lambda >> 1$. We have tried to compute the average and variance of Δ under both hypotheses using appropriate approximations, and then approximate the distribution of Δ using CLT[2] [7] by Normal

[2] Central Limit Theorem

distribution with corresponding averages and variances.

In the appendix we have shown that the average and variance of Δ under hypotheses H_0 and H_1 can be approximately computed using the following formulas.

$$\mu_{\Delta|H_0} = \sqrt{\frac{2}{\pi\lambda}} \tag{12}$$

$$\sigma^2_{\Delta|H_0} = (1-\frac{2}{\pi})\lambda^{-1}2^{-m} \tag{13}$$

$$\mu_{\Delta|H_1} = \lambda^{-1}\mathbf{\mu Q}^T \tag{14}$$

$$\sigma^2_{\Delta|H_1} = \lambda^{-2}(\mathbf{\mu\Lambda\mu^T} + 2^{-m}\mathbf{\sigma^2 Q^T}) \tag{15}$$

Q and **Λ** are the average vector and covariance matrix of the vector random variable $[T_0\ T_1\ \ldots\ T_K]$ for a randomly chosen function from Ω, defined in Section 4. **μ**=[$\mu_0\ \mu_1\ \ldots\ \mu_K$] and $\mathbf{\sigma^2} = [\sigma_0^2\ \sigma_1^2\ \ldots \sigma_K^2]$ where μ_k and σ_k^2 are defined as follows and K=10 is enough for good approximation.

$$\mu_k = \begin{cases} \lambda & k=0 \\ \sqrt{\dfrac{2\lambda}{\pi}} & k=1 \\ \approx (k-1)\lambda & k \geq 2 \end{cases}$$

$$\sigma_k^2 = \begin{cases} 0 & k=0 \\ (1-\dfrac{2}{\pi})\lambda & k=1 \\ k\lambda + (k-1)^2\lambda^2 - \mu_k^2 & k \geq 2 \end{cases}$$

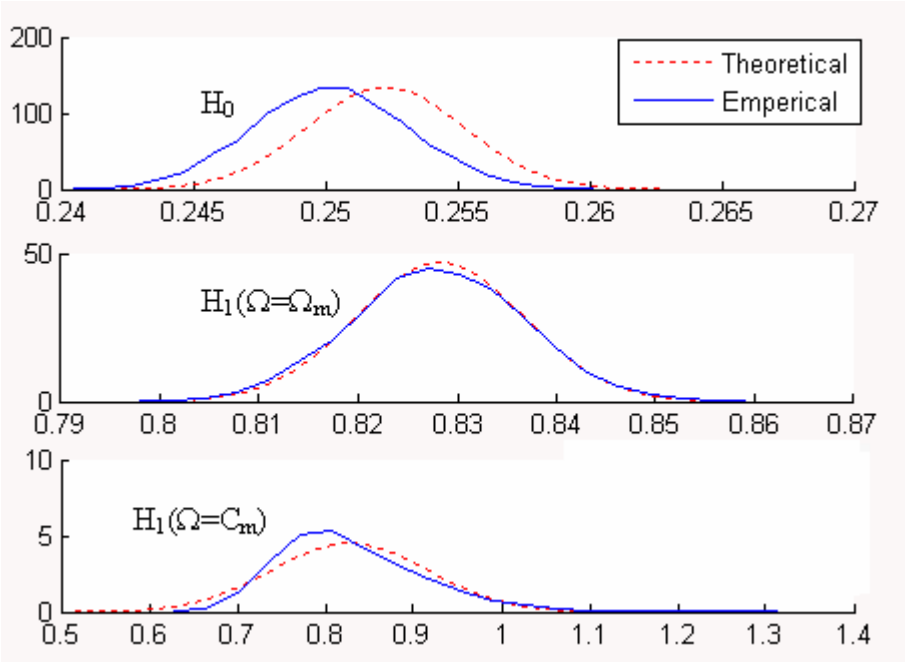
Our simulations show that the above theoretical analysis is very accurate. We have plotted the theoretical and empirical distribution of Δ for λ=10 and m=12 in Figure 1 for three different cases, under hypothesis H_0, under hypothesis H_1 assuming $\Omega=\Omega_m$ and under hypothesis H_1 assuming $\Omega=C_m$. Since the resolution of x-axis is very high in diagrams of Figure 1, the difference between the theoretical and empirical distribution of Δ may be a little eye-catching. However plotting these six diagrams in a unit coordinate, the difference between theoretical and empirical distribution will not be sensible and they seem to be coincident. This shows that as we expected the theoretical approximation is very well. In Figure 2 we have plotted just the empirical distribution of Δ for λ=10 under hypothesis H_0, under hypothesis H_1 assuming $\Omega=\Omega_m$ and under H_1 assuming $\Omega=C_m$ for three different values of m (4, 8, 12). The empirical distributions of Figures 1 and 2 have been estimated over 10,000 randomly chosen sample sequences. Under hypothesis H_1, 100 sample functions from Ω have been randomly chosen to estimate the **Q** vector and **Λ** matrix.

Since $\sigma^2_{\Delta|H_0}$ exponentially decreases when m increases, under hypothesis H_0 the distribution of Δ approaches to a delta function centered on $\mu_{\Delta|H_0} = \sqrt{\frac{2}{\pi\lambda}}$ which is

approximately equal to 0.2523 for λ=10. As we mentioned in Section 4, our simulations show that increasing m, the entries of $\mathbf{\Lambda}$ exponentially approach to zero, $q_0, q_1 \to e^{-1}$ and $\sum_{k=0}^{K} kq_k \to 1$ for both $\Omega=\Omega_m$ and $\Omega=C_m$. Using these results it could be shown that under hypothesis H_1 the distribution of Δ approaches to a delta function centered on $\mu_{\Delta|H_1} \approx (2\lambda + \sqrt{\frac{2\lambda}{\pi}})e^{-1}$ which is approximately equal to 0.8285 for λ=10. See Figures 1 and 2 to know how our results are accurate.

The more m increases, the easier the discrimination between these two hypotheses will be for both $\Omega=\Omega_m$ and $\Omega=C_m$. Since the variance of Δ approaches to zero for $\Omega=\Omega_m$ a bit faster than that for $\Omega=C_m$, distinguishing is a bit easier for $\Omega=\Omega_m$ than $\Omega=C_m$. As m increases this will not be important and shows that choosing C function randomly from C_{32} is approximately the same as choosing it from Ω_{32}.

Figure 1. Theoretical and empirical distribution of Δ for λ=10 and m=12 under hypothesis a) H_0, b) H_1 ($\Omega=\Omega_m$) and c) H_1 ($\Omega=C_m$) on the same coordinate



In hypothesis testing we deal with two error probabilities $p_{miss} = \Pr\{H_0 \mid H_1\}$ and $p_{fa} = \Pr\{H_1 \mid H_0\}$. To be more precise, if we rely on our theoretical analysis, p_{miss} can be computed versus p_{fa} by eliminating γ from the following equations.[3]

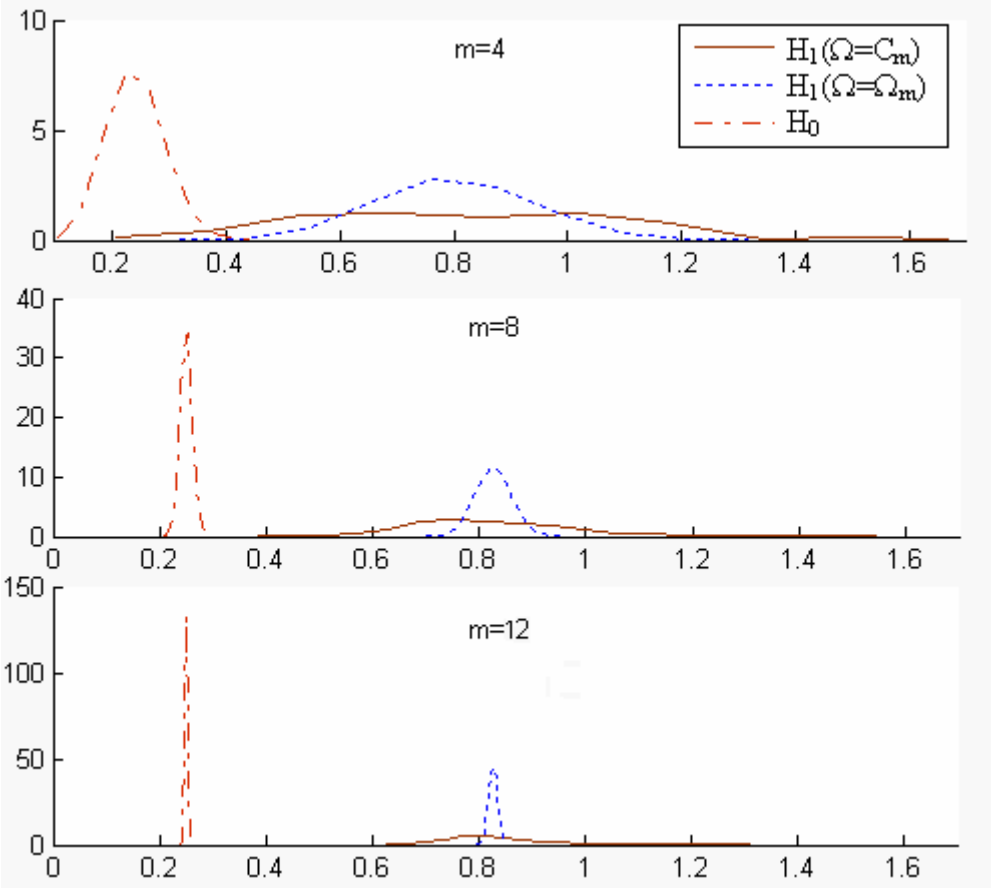
[3] Q function is defined as $Q(z) = \int_z^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$.

$$p_{fa} = \Pr\{\Delta > \mu_{\Delta|H_0} + \gamma\sigma_{\Delta|H_0} | H_0\} = Q(\gamma) \quad (16)$$

$$p_{miss} = \Pr\{\Delta < \mu_{\Delta|H_0} + \gamma\sigma_{\Delta|H_0} | H_1\} = Q\left((\mu_{\Delta|H_1} - (\mu_{\Delta|H_0} + \gamma\sigma_{\Delta|H_1}))/\sigma_{\Delta|H_0}\right) \quad (17)$$

We will refer to these equations to analyze our attack in Section 5.

Figure 1. Empirical distribution of Δ for λ=10 under hypothesis H_1 (Ω=Ω_m), H_1 (Ω=C_m) and H_0, for a) m=4, b) m=8 and c) m=12 on the same coordinate



5 Attack Procedure

In this section we explain how to use the ability to distinguish the distribution of C(x) from the uniform distribution to develop a divide and conquer attack on ABC to find the initial value of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers. In fact this is not a new attack and can be considered as a generalization of correlation attack introduced in [4] by Siegenthaler. The aim of correlation attack introduced by Siegenthaler is to find the initial state of a binary LFSR of length L, given the noisy output sequence of a BMSC[4] when the output sequence of the LFSR is applied to the input of this channel. As the first N bits of the output sequence of an LFSR is a codeword of the corresponding truncated cyclic linear code of given LFSR, the problem is essentially a decoding problem. Siegenthaler solves this problem

[4] Binary Memoryless Symmetric Channel

using ML decoding and shows that in order to successfully finding the initial state of the LFSR, the minimum required output length of the given noisy sequence, N, is determined by the error probability of the corresponding channel [4]. In this case, ML decoding is performed by searching over all 2^L possible initial states for the LFSR and choosing one that its corresponding output sequence has the least hamming distance from the given noisy sequence, assuming that the channel error probability is less than one half.

Looking into equation (9), the output expression of the ABC, it is clear that $\{y_n\}$ can be considered as the noisy version of $\{z_n^0\}$, where $\{C(x_n)\}$ is treated as the noise sequence. We are going to find the correct value of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers by exhaustive search over all 2^{63} possible values for them. Corresponding to each assigned value $\hat{\mathbf{z}}^1$ and $\hat{\mathbf{z}}^0$ to the registers $\mathbf{z}^1$ and $\mathbf{z}^0$, we denote the states sequence of these registers by $\{\hat{z}_n^1\}_{n=0}^{\infty}$ and $\{\hat{z}_n^0\}_{n=0}^{\infty}$. For each assigned value to the registers $\mathbf{z}^1$ and $\mathbf{z}^0$, we compute the sequence $\{w_n\}_{n=1}^{\infty}$, where $w_n = y_n - \hat{z}_n^0$ for $n \geq 1$.

If we are trying the correct initial state of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers we have $\hat{z}_n^0 = z_n^0$ which shows that $w_n = C(x_n)$. So in regard to good statistical properties of $\{x_n\}$, the sequence $\{w_n\}_{n=1}^{\infty}$ seems as a filtered uniform sequence, where the filter function has been chosen randomly from C_{32}. If we are trying an incorrect initial state of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers, we have $w_n = y_n - \hat{z}_n^0 = z_n^0 - \hat{z}_n^0 + C(x_n)$. Because of good statistical properties of LFSR output sequences, $\{\hat{z}_n^0\}$ and $\{z_n^0\}$ can be regarded as mutually independent purely random sequences. In this case the sequence $\{z_n^0 - \hat{z}_n^0\}$ also has very good statistical properties and can hide the non-uniform distribution of $\{C(x_n)\}$ very well. So it is reasonable to consider the sequence $\{w_n\}_{n=1}^{\infty}$ as a uniform sequence for an incorrect guess of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers.

Therefore we are dealing with two hypotheses $\hat{H}_1$ and $\hat{H}_0$, respectively corresponding to a correct initial state and an incorrect one for $\mathbf{z}^1$ and $\mathbf{z}^0$ registers. Given a sequence of length N of the output sequence, that is $\{y_n\}_{n=1}^{N}$, we are going to find the most likely candidates for the initial value of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers. In order to successfully finding of the initial value of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers, the error probability $p_{fa} = \Pr\{H_1 \mid H_0\}$ must be in the order of 2^{-63} while the error probability $p_{miss} = \Pr\{H_0 \mid H_1\}$ has been set to a reasonable value, e.g. 0.01 or 0.1. The rationale behind this expression is that altogether there are 2^{63}-1 incorrect choices for the initial value of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers. As p_{fa} has been set to 2^{-63} we expect that in average one of these incorrect choices be in the candidate set for the initial value of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers while with a good probability the correct choice has not been missed.

Computation of the minimum required output length of the key stream generator to apply a successful attack is under investigation. At this point using our theoretical results in Section 4, we show that using 10×2^{32} words of the output sequence, the attack is successful.

As $Q(9) \approx 2^{-63}$, setting γ=9 in equations (16) satisfies the required p_{fa}. Since computing p_{miss} using equation (17) needs to know $\mathbf{Q}$ and $\mathbf{\Lambda}$ for C_{32} which take a long time to estimate them ($O(2^{32})$), we have not computed it, but we practically expect to be zero for λ=10. Note that λ=10 is the minimum value of $\lambda >>1$ which our theoretical analysis is valid. Since for each possible initial state of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers we have to process 10×2^{32} words, the time and data complexities of this phase are 10×2^{95} simple word operations and 10×2^{32} words, respectively.

In the rest of this section we explain how to attack the whole cipher after finding the initial state of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers. Having found the initial value of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers, the attacker can search over all possible initial value of register $\mathbf{x}$ and constant values $\mathbf{d}_0$ and $\mathbf{d}_1$. Knowing the values of these three registers and finding the initial values of $\mathbf{z}^1$ and $\mathbf{z}^0$ registers at the previous step, the attacker can construct a linear equation for 33 unknown parameters $\mathbf{e}$, $\mathbf{e}_1$, $\mathbf{e}_2$, …, $\mathbf{e}_{32}$ according to (9) and (3). The coefficient of these equations come from GF(2). Using slightly more than 33 equations, we can be sure that we have 33 linearly independent equations which reveals the values of these unknown parameters. After finding the values of the parameters $\mathbf{e}$, $\mathbf{e}_1$, $\mathbf{e}_2$, …, $\mathbf{e}_{32}$, the attacker can check the correctness of her/his guess by comparing a few other output words of the output sequence. Since $\mathbf{d}_0 \equiv 1 \pmod 2$ and $\mathbf{d}_1 \equiv 0 \pmod 4$ the attacker must try $2^{32+31+30}$ guesses for unknown values of $\mathbf{x}$, $\mathbf{d}_0$ and $\mathbf{d}_1$, and then for each guess solves a linear system of 33 unknowns. Therefore, the time complexity of this step is $33^3\times2^{93}\approx 2^{108}$ simple word operations and data complexity is a few words.

In order to summarize the attack results, it can be said that the time and data complexities for breaking the whole cipher are 2^{108} and 10×2^{32}, respectively.

6 Conclusion

In this paper we proposed a divide and conquer correlation attack on ABC stream cipher. We showed that C function in ABC is slightly better than a randomly chosen function over GF(2^{32}). Using this great weakness of C we proposed a correlation based divide and conquer attack which finds the initial values of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers by searching over all 2^{63} possible choices for them in time complexity of 10×2^{95} simple word operations using 10×2^{32} output words. Although the attack is performable using less data and time complexities, we just theoretically analyzed this amount of data.

Assuming that the initial values of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers are available, we suggested to find the values of $\mathbf{e}$, $\mathbf{e}_0$, $\mathbf{e}_1$, … and $\mathbf{e}_{31}$ registers by searching over all 2^{93} possible initial values for $\mathbf{x}$, $\mathbf{d}_0$ and $\mathbf{d}_1$ registers which needs 2^{108} simple word operations and a few words of the output sequence. Therefore the total time and data complexity of our attack for breaking the whole cipher are 2^{108} simple word operations and 10×2^{32} words, respectively.

It is worth investigating the possibility of

1. Computing the minimum required output length for successfully finding the initial values of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers.

2. Finding the initial values of $\mathbf{z}^0$ and $\mathbf{z}^1$ registers using some methods similar to fast correlation attack techniques.
3. Finding (Estimating) the values of $\mathbf{e}$, $\mathbf{e}_0$, $\mathbf{e}_1$, … and $\mathbf{e}_{31}$ just using the distribution of C function without searching over $\mathbf{x}$, $\mathbf{d}_0$ and $\mathbf{d}_1$ registers.

References

1. V. Anashin, A. Bogdanov, I. Kizhvatov. "ABC: A New Fast Flexible Stream Cipher," available at http://www.ecrypt.eu.org/stream/abc.html.
2. http://www.ecrypt.eu.org/stream/
3. A. Klimov and A. Shamir, "A new class of invertible mappings," in: Cryptographic Hardware and Embedded Systems 2002 (B.S.Kaliski Jr.etal., eds.), Lect. Notes in Comp. Sci.,Vol. 2523, Springer-Verlag, 2003, pp.470–483.
4. T. Siegenthaler, "Decrypting a class of stream ciphers using cipher-text only", IEEE Trans. Computers., vol. c-34, no. 1, January 1985, pp. 81-85.
5. H. Vincent Poor, An introduction to signal detection and estimation (2nd ed.), Springer-Verlag New York, Inc., New York, NY, 1994 Beker & Piper
6. H. Beker and F. Piper, Cipher Systems. John Wiley & Sons, 1982.
7. Populis, A., "Probability, Random Variable and Stochastic Process", Mc. Graw-Hill, 1991.

Appendix

Deriving the equations (12-15)

In this section we try to compute the distribution of Δ under both hypotheses H_0, H_1 using appropriate approximations. For $N = \lambda 2^m$, (11) is simplified as

$$\Delta = \lambda^{-1} 2^{-m} \sum_{x=0}^{2^m-1} | \hat{f}_x - \lambda |$$

If $p_x = k/2^m$, $\hat{f}_x$ has Binomial distribution $B(\lambda 2^m, k2^{-m})$, where under the condition $\lambda >> 1$ could well be approximated by Normal $N(k\lambda, k\lambda)$ distribution [6].[5] Using this approximation it can be shown that the average and variance of $| \hat{f}_x - \lambda |$, denoted respectively by μ_k and σ_k^2, are as follows.

[5] The probability mass function of B(n,p) and the probability density function of $N(\mu,\sigma^2)$ are respectively defined by $\{\Pr\{x = k\} = \binom{n}{k} p^k (1-p)^{n-k}, k = 0, 1, 2, ..., n\}$ and $f_x(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$.

$$\mu_k = \begin{cases} \lambda & k = 0 \\ \sqrt{\dfrac{2\lambda}{\pi}} & k = 1 \\ (k-1)\lambda\left(1 - 2Q\left(\sqrt{\dfrac{(k-1)^2\lambda}{k}}\right)\right) + \sqrt{\dfrac{2k\lambda}{\pi}} e^{-\frac{(k-1)^2\lambda}{2k}} \approx (k-1)\lambda & k \geq 2 \end{cases}$$

$$\sigma_k^2 = \begin{cases} 0 & k = 0 \\ (1 - \dfrac{2}{\pi})\lambda & k = 1 \\ k\lambda + (k-1)^2\lambda^2 - \mu_k^2 & k \geq 2 \end{cases}$$

Using the computed values of μ_1 and σ_1^2, the average and variance of Δ under hypothesis H_0 are as follows.

$$\mu_{\Delta|H_0} = \sqrt{\frac{2}{\pi \cdot \lambda}}, \sigma^2_{\Delta|H_0} = (1 - \frac{2}{\pi})\lambda^{-1} 2^{-m}$$

It must be noted that although the random variables $\hat{f}_x$ and $\hat{f}_y$ are not independent, we have considered them as independent random variables in deriving the variance of Δ.

To compute the average and variance of Δ under hypothesis H_1 Let again T_k denote the proportion of outputs of C which have been repeated exactly k times in the output range of it. In addition we suppose T_k is equal to zero for k>K. If we knew the values of T_k, the average and variance of Δ under hypothesis H_1 could be computed as

$$E\{\Delta \mid H_1, T_k's\} = \lambda^{-1} \sum_{k=0}^{K} T_k \mu_k$$

$$Var\{\Delta \mid H_1, T_k's\} = \lambda^{-2} \cdot 2^{-m} \sum_{k=0}^{K} T_k \sigma_k^2$$

Under hypothesis H_1, where the values of T_k's are not known, we can treat them as random variables. Using **Q** vector and **Λ** matrix defined in Section 4, the average and variance of Δ can be computed using a generalization of Random Sum Problem [6] which has been retold here.

Random Sum Problem. Let $\{\mathbf{x}_k\}_{k=1}^{\infty}$ denote a sequence of i.i.d. random variables with average and variance μ_x and σ_x^2, and **n** be a discrete random variable over positive

integers, independent of $\{\mathbf{x}_k\}$, with average and variance μ_n and σ_n^2. The average and variance of random variable $\mathbf{S_n} = \sum_{k=1}^{\mathbf{n}} \mathbf{x}_k$ are equal to:

$$E\{\mathbf{S_n}\} = \mu_n \mu_x, Var(\mathbf{S_n}) = \mu_x^2 \sigma_n^2 + \mu_n \sigma_x^2.$$

Generalized Random Sum Problem. Let $\{\mathbf{x}_k^l\}_{k=1}^{\infty}$, $l = 1,2,...,L$ denote L mutually independent sequence of i.i.d. random variables with average and variance μ_j and σ_j^2, and $\mathbf{n=[n_1, n_2, \ldots, n_L]}$ be a discrete vector random variable over positive integers, independent of $\{\mathbf{x}_k^j\}$, with average vector $\boldsymbol{\mu}_\mathbf{n}$ and variance covariance matrix $\boldsymbol{\Lambda}_\mathbf{n}$. The average and variance of random variable $\mathbf{S} = \sum_{l=1}^{L} \sum_{k=1}^{\mathbf{n}_l} \mathbf{x}_k^l$ are equal to:

$$E\{\mathbf{S_n}\} = \mu_n \mu_x, Var(\mathbf{S}) = \boldsymbol{\mu}_\mathbf{x} \boldsymbol{\Lambda}_\mathbf{n} \boldsymbol{\mu}_\mathbf{x}^\mathbf{T} + \boldsymbol{\sigma}_\mathbf{x}^\mathbf{2} \boldsymbol{\mu}_\mathbf{n}^\mathbf{T}$$

where T denotes vector transposition, $\boldsymbol{\mu}_\mathbf{x} = [\mu_1, \mu_2, ..., \mu_L]$ and $\boldsymbol{\sigma}_\mathbf{x}^\mathbf{2} = [\sigma_1^2, \sigma_2^2, ..., \sigma_L^2]$.

Using above theorem considering just first K+1 significant values of T_k, setting $\mathbf{n}_k = 2^m T_k$ we have

$$\mu_{\Delta|H_1} = \lambda^{-1} \boldsymbol{\mu} \mathbf{Q}^T$$

$$\sigma^2_{\Delta|H_1} = \lambda^{-2} (\boldsymbol{\mu} \boldsymbol{\Lambda} \boldsymbol{\mu}^\mathbf{T} + 2^{-m} \boldsymbol{\sigma}^\mathbf{2} \mathbf{Q}^\mathbf{T})$$

$\boldsymbol{\mu}$=[μ_0 μ_1 ... μ_K], $\boldsymbol{\sigma}^\mathbf{2} = [\sigma_0^2\ \sigma_1^2 \ldots \sigma_K^2]$ and $\mathbf{Q}$ and $\boldsymbol{\Lambda}$ are the average vector and covariance matrix of the vector random variable [T_0 T_1 … T_K] for a randomly chosen function from Ω. For good approximation it is sufficient to take K=10.

Here again we have used the independence assumption of random variables $\hat{f}_x$ and $\hat{f}_y$ to compute the variance of Δ.